Using Concept Maps to More Efficiently Create Intelligence Information Models

THESIS

Christopher E. Coryell, Captain, USAF

AFIT/GCE/ENG/07-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCE/ENG/07-03

Using Concept Maps to More Efficiently Create Intelligence Information Models

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Christopher E. Coryell, B.S.C.E.

Captain, USAF

March 2007

AFIT/GCE/ENG/07-03

Using Concept Maps to More Efficiently Create Intelligence Information Models

Christopher E. Coryell, B.S.C.E.

Captain, USAF

Approved:

/signed/	06 Mar 2007
Maj Christopher B. Mayer, (Chairman)	date
/signed/	06 Mar 2007
Maj Scott R. Graham (Member)	date
/signed/	06 Mar 2007
Lt Col Timothy J. Halloran (Member)	date

AFIT/GCE/ENG/07-03

Abstract

Information models are a critical tool for U.S. intelligence agencies to allow their customers to quickly and accurately comprehend their products. The Knowledge Prepositioning System (KPS) is the standard repository for information models at the National Air and Space Intelligence Center (NASIC). The current approach used by NASIC to build a KPS information model is laborious and costly. Intelligence analysts design an information model using a manual, butcher-paper-based process. The output of their work is then entered into KPS by either a single NASIC KPS "database modeler" or a contractor (at a cost of roughly $100K to the organization).

This thesis proposes a tool-supported approach to allow intelligence analysts to create KPS information models with almost no database modeler or contractor support. Our approach allows analysts to express an information model as a *concept map*, an analyst-understandable model of an intelligence domain. An existing tool, CmapTools [6], supports the analyst-in-the-loop process of concept map creation. A completed concept map is automatically converted into KPS by a prototype tool, called Cmap Conversion for KPS, created as part of this work. We restrict, to a reasonable degree, how analysts express concept maps within CmapTools to ensure that automatic conversion into KPS is possible.

We validated our approach using a representative NASIC-provided KPS information model: performance of fixed-wing aircraft. Using our tools, a new information model was constructed in 4 hours and 20 minutes, a 89% improvement over the 40 hours estimated by NASIC to complete the same task using their existing approach. For this representative information model, NASIC estimates our approach would save them roughly $200K.

Acknowledgements

I would like to thank my faculty advisor, Lt Col Timothy Halloran for his assistance during this thesis effort. I would also like to recognize my committee members, Maj Chris Mayer and Maj Scott Graham, for their efforts. Additionally, I would like to express my gratitude to Ms. Sharon Cain and Mr. James Homer of NASIC for their input. Finally, to my wife and daughters, thank you for putting up with a "disappearing" dad from time to time to make this possible.

Christopher E. Coryell

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
KPS	Knowledge Pre-positioning System	iv
IHMC	Institute for Human and Machine Cognition	2
ERD	Entity Relationship Diagram	4
CSIM	CMap Supported Information Modeling	6
IHMC	Institute for Human and Machine Cognition	8
NASIC	National Air and Space Intelligence Center	12
XML	Extensible Markup Language	14
SAVANT	Systematic Architecture for Virtual Analytic Net-centric Threat Information	15
KPS	Knowledge Pre-positioning System	16
ERD	Entity Relationship Diagram	16
FWA	Fixed-Wing Aircraft	19
SAX	Simple API for XML	37

Using Concept Maps to More Efficiently Create Intelligence Information Models

I. Introduction

1.1 The Problem

Information models are a critical tool for U.S. intelligence agencies to allow their customers to quickly and accurately comprehend their products. The Knowledge Prepositioning System (KPS) is the standard repository for information models at the National Air and Space Intelligence Center (NASIC). Information models are a proven technique to pass ideas between individuals of different specialties [4]. An information model is a common way to express data. It can be used to bridge the gap between any number of differing vocabularies as long as the elements of the model are defined in such a way as to be clear to each concerned party what the model represents.

At NASIC, communication is required between intelligence analysts and database modelers to create a KPS information model. When describing a set of data, intelligence analysts speak in terms of products, background information and conclusions; database modelers use terms such as entities, attributes and relationships. These separate specialized vocabularies can cause unneeded and costly delays as the individuals must pass information between one another in order to accurate depict the analyst's knowledge as a KPS information model.

To help solve this problem, an approach is needed to ensure that the relevant information is not only captured by the analyst, but disseminated and stored in such a way that it is easily accessible when needed and can be reconstructed to provide an accurate representation of the intelligence product. Creating this approach is the focus of our work and is motivated by NASIC's need to empower intelligence analysts with the ability to create KPS information models without a KPS-savvy database modeler being a bottleneck. From the larger perspective, the problems with their current approach to KPS information map creation are becoming a hindrance to the timely execution of NASIC's mission.

1.2 This Thesis

This thesis proposes a tool-supported approach for the creation of information models by intelligence analysts that can be automatically translated into a form that can be directly imported into KPS. This approach will create a type of information model called a concept map using an existing tool called CmapTools created by the Institute for Human and Machine Cognition (IHMC). The concept maps will then be converted into files usable by the NASIC data base modeler to populate the intelligence database using a tool we developed as part of our work named CMap Conversion for KPS. A case study using data provided by NASIC indicates that our approach has the potential to save both time and money and can aid NASIC in providing timely, accurate intelligence to the Department of Defense and our allies.

1.3 KPS Information Model Creation Today

This section briefly describes the current approach NASIC uses to create information models. An activity diagram for this approach can be seen in Figure 1.1. The process begins when an intelligence analyst recognizes information that needs to be represented in an information model. At this point the intelligence analyst must decide whether or not the information is time sensitive. If it is not, an appointment has to be made with the database modeler. It may take up to two weeks for the database modeler to have the opportunity to meet with the intelligence analyst.

If the intelligence analyst decides the information is time sensitive, they must contact a database designer; generally a contractor who has the ability to write to the database. This adds a cost of roughly $100K to the process. The database modeler, possibly a contractor, will then sit down with the intelligence analyst and the two work together to formulate an information model that can be parsed into acceptable entities for use in the database. After the information model has been created, the

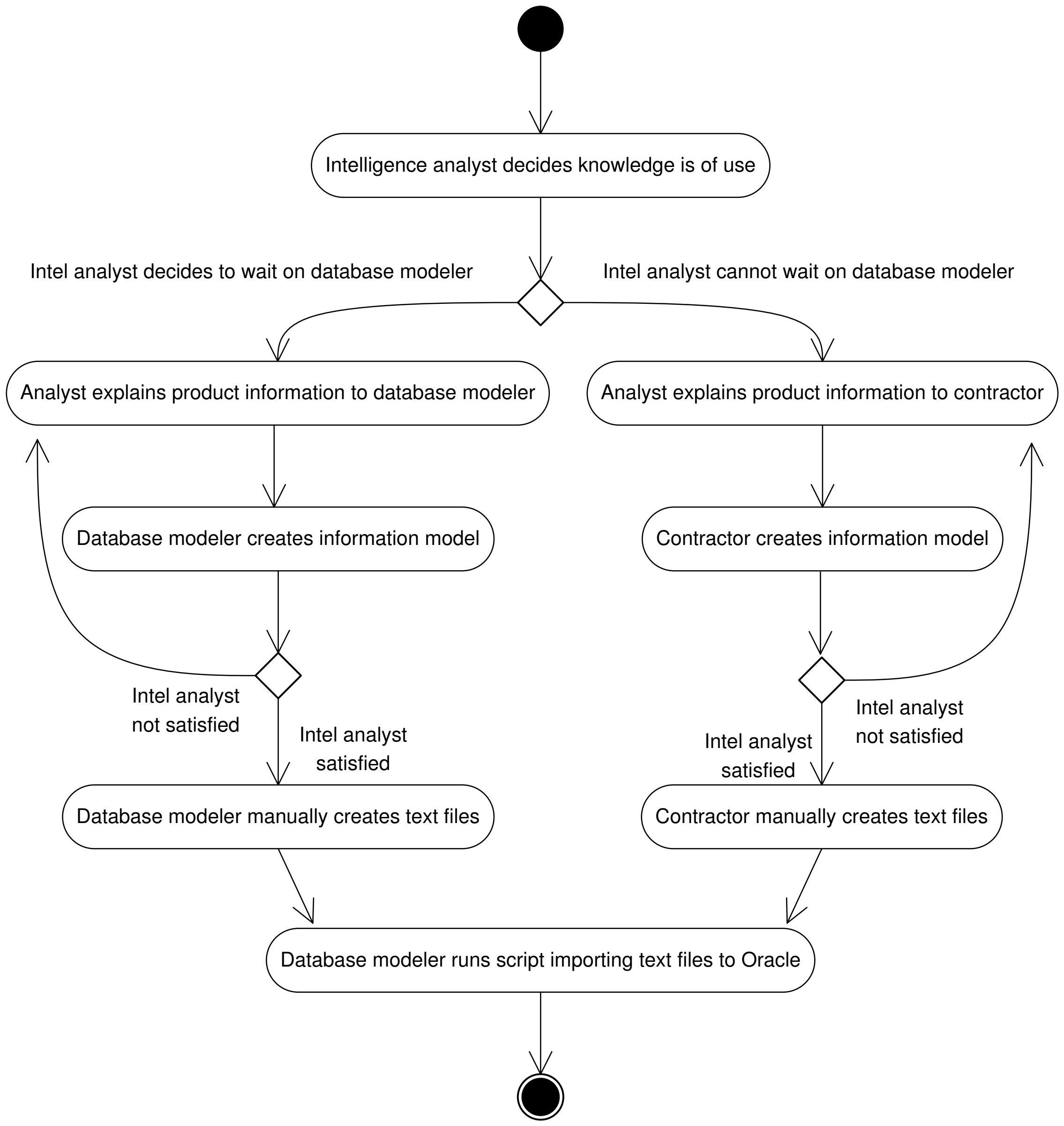

Intelligence analyst decides knowledge is of use
Intel analyst decides to wait on database modeler
Intel analyst cannot wait on database modeler
Analyst explains product information to database modeler
Analyst explains product information to contractor
Database modeler creates information model
Contractor creates information model
Intel analyst not satisfied
Intel analyst satisfied
Intel analyst satisfied
Intel analyst not satisfied
Database modeler manually creates text files
Contractor manually creates text files
Database modeler runs script importing text files to Oracle

Figure 1.1: Activity diagram of NASIC's current approach.

database modeler then creates verbose text files of the information so that existing scripts can be used to read in the data and import it into NASIC's database.

1.3.1 Problems. There are several problems with the current approach. First, intelligence analysts may not have collected the breadth of information required by KPS. When the intelligence analyst sits down with the database modeler to create an information model, the intelligence analyst comes to realize that a complete information model requires a great deal of facts and these particulars are not at the forefront of the intelligence analysts mind.

Second, intelligence analysts do not understand Entity Relationship Diagram (ERD) [9] notation or semantics used by KPS. For example, an intelligence analyst knows that a report about a fixed wing aircraft needs to contain information about the combat capabilities of the aircraft. However, the database modeler has to explain that both the aircraft and its capabilities are represented as entities in the database due to the large amount of information that can be captured about the combat capability. At times, this confusion about ERD requirements causes "bad" information models to be uploaded to the database. When these models are identified, additional time is needed for the intelligence analyst and database modeler to confer and correct the mistakes.

Third, the process used by intelligence analysts to gather their requirements is an ad-hoc, often butcher-paper-based, process. Therefore, the process of information model creation can be time intensive as the intelligence analysts attempt to communicate and, often times, formulate their requirements to the database modeler. According to Ms. Sharon Cain, the sole NASIC database modeler, it currently takes approximately 40 hours to manually create an information model.

Fourth, the availability of only a single NASIC database modeler can cause time delays. In effect, the database modeler becomes a "bottleneck" in the process. Making the situation worse, KPS information map creation is not the only task assigned to the NASIC database modeler—other duties can and do preempt work on KPS information models. Therefore, instead of an intelligence analyst being able to put their ideas into production as soon as they are organized, a waiting game has to be played until time can be scheduled with the database modeler. This can cause delays of up to a month between the intelligence analysts conception of the intelligence product and its inclusion into the database. These setbacks can drastically reduce the relevance of the intelligence information. Due to these scheduling difficulties, in lieu of the database modeler, a contractor is often used to co-create the information models adding an estimated $100K to the cost of model creation.

Fifth, the database modeler must translate the information models into a very exacting text file format before the information model can be loaded into KPS. KPS currently relies on basic scripts to parse text files to create the necessary database entries. These text files are very difficult to read and therefore, syntactical mistakes are common during manual creation.

Finally, with the intelligence analyst's limited understanding of the ERD, mistakes in the newly created models are often not uncovered until after the information model is put into production, sometimes taking weeks to be discovered. This creates the possibility of mission planning based on incorrect intelligence. When mistakes are finally noted, the scheduling conflicts discussed above once again come into play.

Overall, NASIC estimates that the creation cost of each KPS information model is roughly $350K.

1.4 Improving KPS Information Model Creation

A major portion of the delay during information model creation is the nonexistent mapping between analyst ideas and the ERD models needed for the database. If a process existed to more precisely capture the intelligence analysts knowledge in a manner understandable to the analyst, it would ease the database modelers' efforts to map the products to their ERD counterparts.

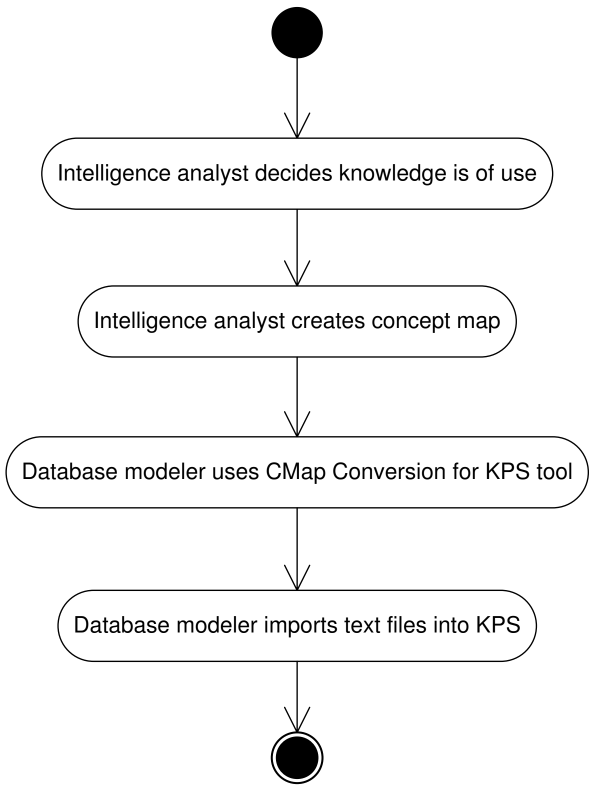
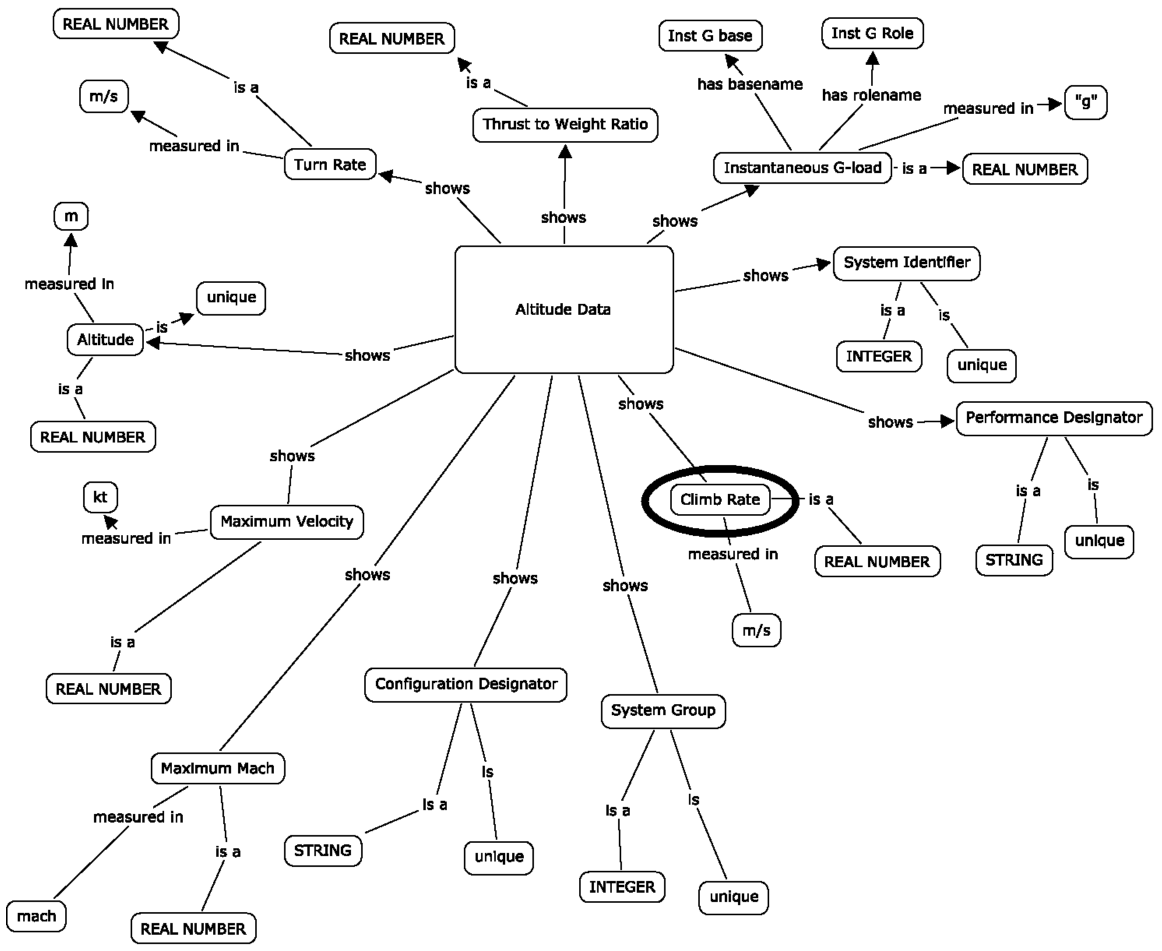

Intelligence analyst decides knowledge is of use
Intelligence analyst creates concept map
Database modeler uses CMap Conversion for KPS tool
Database modeler imports text files into KPS

Figure 1.2: Our proposed information model creation approach.

Intelligence analysts have the choice of waiting behind the bottleneck of the database modeler or paying the cost of a contractor. Allowing them the opportunity to create the information model themselves removes both of these obstacles.

Manually creating obtuse text files adds unnecessary cost, time delays and database modeler frustration. The original intent behind the text files was to make them easy for a script to parse; not for user readability. A tool that could automatically create these files from the information model would save both time and money.

We propose that properly preparing intelligence data for inclusion into the KPS can be streamlined via the use of an approach we call *CMap Supported Information Modeling* (CSIM) using the steps in Figure 1.2 and described further below.



```
<concept id="1165336643244_205225960_469"
        label="Climb Rate"
        short-comment="The maximum sustainable rate of climb at the
                      specified takeoff total gross weight and
                      altitude at maximum power."
        long-comment="" />
```

Figure 1.3: (top) An example concept map that describes the Altitude concept. (bottom) The XML representation of the Climb Rate concept—Climb Rate is circled in the Altitude concept map.

1.4.1 Intelligence Analyst Decides Knowledge is of use. In essence, the intelligence analyst deems there is a use for a new intelligence product. This decision starts the process of creating a new KPS information model.

1.4.2 Intelligence Analyst Creates a Concept Map. The tool we have chosen for intelligence analysts to create concept maps is CmapTools [6]. This application was developed by the Institute for Human and Machine Cognition (IHMC). It was selected for use in our proposed approach by NASIC due to its cost—it is freely available—and ease of use. Additionally, NASIC has close ties with IHMC, which will allow NASIC to influence future builds of the CmapTools software.

Instead of the current ad hoc process, an intelligence analyst will use CmapTools to create an information model of their intelligence data. This information model, called a concept map, will be a graphical representation of the intelligence analyst's intelligence product. In a concept map, boxes represent the various *concepts* of the map. Concepts are connected with labeled lines which are the *linking phrases* of the map. An example concept map for the Altitude concept is shown in Figure 1.3. The Climb Rate concept, which is circled in Figure 1.3, has its XML representation shown at the bottom of the figure.

When the analyst is satisfied that the concept map accurately depicts the relevant intelligence data, CmapTools can export the concept map as an XML document. CmapTools can use these files to reconstruct concept maps; for example, an employee in Dayton creates a concept map and places the XML in a company server. This XML is then accessed by an employee in Seoul and CmapTools creates an exact representation of the original concept map from the XML. However, for use in our work, the XML file representation of a concept map will be used to create the text files required by KPS.

1.4.3 Database modeler Uses CMap Conversion for KPS tool. The database modeler will use the CMap Conversion for KPS tool (developed as part of this work) to convert the XML concept map and create the necessary text files to upload the data into KPS.

1.4.4 Database modeler Imports Text Files into KPS. The output from the CMap Conversion for KPS tool is imported into KPS using pre-existing scripts on the NASIC network. At this point the new information model is available for use in KPS.

Our proposed tool-supported approach partially solves the problems of the current approach. First, the use of concept maps allows analysts to systematically define their desired intelligence product. In order to properly represent their knowledge in a concept map, intelligence analysts must think critically about the information they are trying to communicate. They must compartmentalize their understanding into main ideas and attributes that describe those ideas. We acknowledge that this will take more time up front than the current approach, however, taking this additional time to create the concept maps will also give the intelligence analysts a feeling of ownership over the ideas they contain. This point of view will foster a desire in the intelligence analysts to make their concept maps as complete as possible and this, in turn, will result in better intelligence products.

Second, our proposed approach removes the database modeler scheduling conflicts noted previously. Concept maps can be created when the intelligence analyst is ready; not when the database modeler is available.

Third, the need for a contractor to accelerate the process is removed. Instead of the many hours required to manually create an butcher-paper-based information model into the text files required by KPS, it will take the database modeler only seconds to use our CMap Conversion for KPS tool to convert the concept map into the necessary text files.

Fourth, our proposed approach will result in fewer mistakes. No tool can be mistake free, however, as intelligence analysts grow to better understand concept maps and the process of creating them, their products will better reflect the ERD concepts behind them ensuring that the intelligence that goes out in production will accurately represent the knowledge of the intelligence analyst.

Fifth, much of the ambiguity of the current approach is removed. Since our CMap Conversion for KPS tool is used to convert the concept map into a KPS information model. To allow this automated conversion, we mandate the use of common keywords to represent particular concepts. This ensures that the intent of the intelligence analyst is accurately communicated in the resulting KPS information model.

Finally, changes to the information models are simplified. With no scheduling difficulties to slow the process down, a change can be made with very little cost. A few minutes to make the change on the original concept map and export the XML document is all it takes. The updated data will be waiting and ready when the database modeler next pushes data to KPS.

1.5 Validation

We validated our claim that our proposed approach is an improvement upon the current approach using a data set provided by NASIC. Our approach reduced the total time required to build an information model and prepare the text files for import to the database 89% percent from 40 hours to 4. Additionally, by removing the need for a contractor, we were able to significantly lower the cost for creation of the information model from $350K to less than $150K according to NASIC estimates.

1.6 Outline

The remainder of this document is organized as follows:

- Chapter II, "Definitions and Prior Work," provides relevant definitions to our work and details our NASIC's prior attempts to solve this problem.
- Chapter III, "Tool Use," describes how to use tools to create a concept map and automatically convert it into a KPS information model.

- Chapter IV, “Tool Engineering,” describes the design and implementation of the CMap Conversion for KPS tool.
- Chapter V, “Validation,” describes a case study we performed on a data set received from NASIC using our approach. This chapter discusses the strengths and weaknesses observed during this case study.
- Chapter VI, “Conclusion,” summarizes our results and covers possible future work.

II. Definitions and Prior Work

This chapter defines key terms and discusses previous attempts by the National Air and Space Intelligence Center (NASIC) to improve their current approach to information model creation for KPS.

2.1 Definitions

In this section we will define all key terms and ideas used in both the formation and completion of this work.

2.1.1 Information Models. An information model is an organizational framework that is used to categorize information resources. Information models are used extensively today by both military and civilian companies as a way to pass information easily and succinctly between employees of varying specialties. Simply put, an information model is a common way to represent data so that it can be understood and disbursed in such a way as to ensure that useful information can have the widest possible dissemination. Information models are a critical tool for U.S. intelligence agencies to allow their customers to quickly and accurately comprehend their products. However, finding information models that can be understood by all levels of the organization is tricky.

In [4], Kostur speaks to these struggles. She states, "many authors struggle with modeling their content; they have difficulty describing their structure semantically, they have difficulty visualizing structure within and across information products, and they have difficulty understanding the technology well enough to know what type of information to include in the models."

She continues to discuss the problems customers have in finding the content that they need and the rising costs of content creation. These problems can be resolved by correct use of information modeling. Concept mapping, discussed below, is an example of this idea.

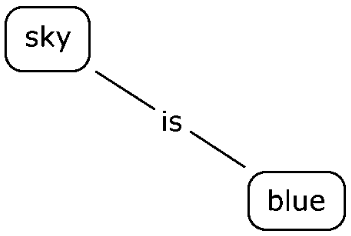


Figure 2.1: A simple concept map of the relationship: "sky is blue."

2.1.2 Concept Maps. Introduced by Novak and Gowin [7], the concept map is a graphical notation that enables knowledge expression in easily understood forms. A concept map is comprised of two separate yet interconnected entities; concepts and the relationships between pairs of concepts, called links. The concepts are referenced as nouns, with the relationships usually consisting of verbs, forming propositions or phrases for each pair of concepts [2]. A very simple concept map, shown in Figure 2.1, would consist of two concept nodes, such as "sky" and "blue." These would then be connected by a directed arc representing the relationship "is." The entire map would then represent the fact that the "sky is blue."

Another characteristic of concept maps is that concepts are represented in a hierarchical fashion with the most inclusive concepts at the top of the map and more specific concepts arranged hierarchically below [5]. This makes concept maps ideal to describe a particular intelligence product. The product can be represented as the root of the map. It can then be fully described by as many "child" nodes as needed to capture the necessary details. This flexibility allows a concept map to be as inclusive or exclusive as the creator of the map desires.

Concept maps are able to use cross-links that define relationships between different domains of the concept map [5]. This capability allows several different products to be described on the same concept map detailing their relationships to one another.

The visual nature of concept maps and their use of simple words or phrases to describe ideas make them an easy notation for intelligence analysts to learn. When analysts realize that their intelligence products can be generalized to one all-encompassing thought, they can then detail their desired product via related "child" concepts.

2.1.3 CmapTools. The CmapTools software is an application created and maintained by IHMC. NASIC has close ties with the IHMC and, after mentioning their prior unsuccessful attempts to create functional information models, IHMC introduced them to the *CmapTools* application. The following characteristics of *CmapTools* [3] make it attractive as an information model creation tool:

- Easy to learn: The process of creating a concept map with CmapTools is very intuitive and an experienced computer user will have no problems creating new concept maps with little or no training. However, CmapTools capability goes far beyond what we made use of in the CSIM approach giving us a large capacity for upgrading the approach at a later date.
- Collaboration and sharing: The CmapTools application allows for concept maps to be shared and synchronously modified. This capability allows the information models to encompass a greater pool of employee knowledge as many different users can collaborate on a singe concept map.
- NASIC can influence tool evolution: Due to NASIC's close working relationship with the IHMC, the users of the CmapTools application will be able to speak directly with the designers of the tool. As such, they will be able to influence modifications and add-ons to the software as needed.
- Cost: The CmapTools software is a freely available software package that can be downloaded from the Internet.
- XML support: As we have discussed, CmapTools has the ability to store a concept map as XML. This capability enables our automatic conversion of the concept map into KPS data files.

2.1.4 XML. The Extensible Markup Language (XML) is a general-purpose markup language used to create special purpose markup languages, which are capable of describing data in any way in which the author designs. More simply, XML is a customizable language used to represent data. CmapTools includes a utility that will export the Concept Maps to a version of XML the creators of CmapTools call Concept Mapping Extensible Language (CXL). The CXL language was designed specifically for concept maps created with CmapTools. It describes all the layout and style definitions as well as any resource links (images, URLs, other concept maps, etc.) that are included in the concept map. As CXL is a stylized version of XML and our work does not use any other versions of XML, any mention of XML can be assumed to be CXL for the remainder of this document.

2.1.5 Systematic Architecture for Virtual Analytic Net-centric Threat Information (SAVANT).

To understand NASIC's desire for the migration of intelligence information into an information model, it is necessary to understand the steps they have taken to create a workspace environment conducive to correctly catalog and disburse intelligence information. To this end, NASIC created the Systematic Architecture for Virtual Analytic Net-centric Threat Information (SAVANT) program. This program was developed by NASIC in 2003 to provide for 21st century transformation and modernization to analyze, predict, capture, convey and disseminate intelligence information. Goals of the SAVANT program were:

- Position, catalog and access all NASIC information.
- Improve integration across mission areas.
- Provide access to information at appropriate security level on demand
- Improve methods to capture and articulate threat information.
- Reduce costs through a standardized digital production approach.
- Enable improved "Machine-to-Machine" communication capability.
- Support various Intelligence Community initiatives.

The SAVANT program encompasses various applications, however, the portion of SAVANT that we are most concerned with is the Knowledge Pre-positioning System (KPS).

KPS provides a collection of tools and services for analysts to capture analysis data and organize knowledge in a structured manner. Unfortunately, KPS fell short of its intended goal and it is in this area that our approach seeks to better the organization. The current approach under KPS is very generalized in its approach and, as such, there are many differing ideas of what an intelligence product and corresponding information model should look like. Our approach proposes a single method for creating information models thereby limiting the possibility of misunderstanding the intelligence analysts intent.

2.1.6 Entity Relationship Diagram (ERD). A major flaw with the SAVANT program is the inability of intelligence analysts to format their knowledge into a form that is easily translatable into an ERD. An ERD is another example of an information model. As one would expect from the name, the ERD models the relationships between the various entities in a database. Database modelers and database administrators use ERDs to accurately display the design and structure of their databases.

The basic ERD consists of three types of objects: entities, relationships, and attributes [9].

- Entities represent the central objects that information is being collected about. An entity can typically be thought of as the subject of a sentence and thus can be expressed by a noun.
- Relationships represent real-world associations among one or more entities. The relationships commonly refer to the connectivity between entities; that is, whether the relationship is one-to-one, one-to-many, or many-to-many.
- Attributes describe the characteristics of the entities. Just as entities can be expressed as nouns, the attributes can be thought of in terms of adjectives. If

the simple concept map in Figure 2.1 is represented an ERD, “sky” would be an entity while “blue” would be an attribute of the “sky” entity.

2.2 Prior Work

This section will discuss prior attempts by NASIC to find a more efficient approach to KPS information map creation.

2.2.1 Attempts at a Standard Information Model. Recognizing the need for a standard information model for KPS to be shared throughout NASIC, there have been several attempts to create a model that would be used across the different directorates. To assist in this process, NASIC has hired representatives from Northrop Grumman who understand the ERD concept. However, different teams of contractors were assigned to each directorate and, due to poor communication between these teams, stove pipe systems have begun to appear exacerbating the problem that Northrop Grumman was employed to correct.

2.2.2 Model Creation Tools. In addition to our proposed use of *CmapTools*, NASIC’s planning branch has tested several other modeling applications. Each of these tools were discarded for the reasons detailed below:

- Visual Thought: Visual Thought is a diagramming and flow chart tool made by Confluent. It is able to be used across platforms and is very flexible. However, contractor intervention is necessary to convert the Visual Thought models into files usable by KPS and at a price of $950 per license, this method is cost prohibitive.
- Microsoft Visio: Visio was less expensive than Visual Thought, but also more restrictive. It is intended for use on a PC, and although it does support customization, it is not a simple process. Again, it also required a contractor for file conversion. This method was deemed unusable as NASIC planners do not believe that intelligence analysts will like the cumbersome nature of the required

Visio customization. Additionally, they are attempting to move beyond the need to use contractors for file conversion.

- Microsoft PowerPoint: As PowerPoint is intended to build presentations, it does not include the necessary customization options needed in order to accurate represent a information model of this complexity. The lone reason it was used was cost as NASIC current holds a site license for the software. Also, as before, it required a contractor to convert the final model into KPS usable files. PowerPoint was discarded after a very brief trial as it was apparent that this application was not intended to produce intelligence information models.

III. Tool Use

This chapter describes the proposed approach to be used to move information from the mind of the intelligence analysts into text files used by the database modeler to populate the database. This approach can be divided into three steps:

1. Creation of the concept map.
2. Exportation of the concept map to XML.
3. Execution of CMap Conversion for KPS to create KPS text files.

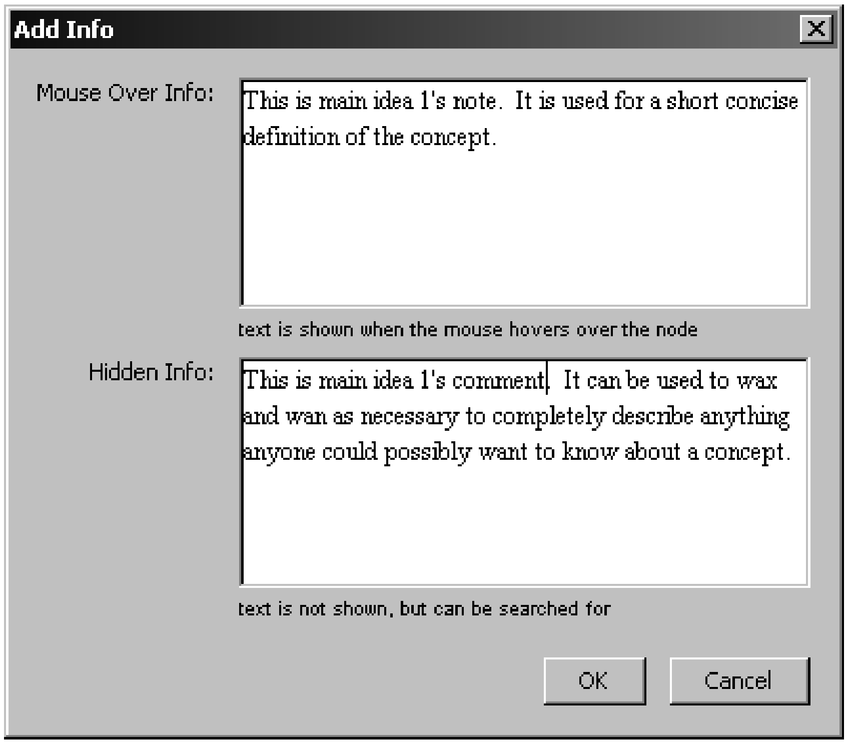
3.1 Concept Map Creation

Creating an accurate, complete, correctly-stylized concept map is the most important step in this approach. Unless care is taken in the creation of the concept map, the intent of the intelligence analyst will not be correctly transferred by the remaining steps effectively rendering the entire approach useless. Various keywords are used throughout concept map creation and are looked for by the remaining steps of the approach so spelling and correct usage is imperative. This portion will be completed entirely within the CmapTools application.

3.1.1 Create new Concept Map. From the menu, select File, then New CMap.

3.1.2 Create new Concept. Double-click in the middle of the Untitled canvas to create a new concept. A new concept will appear at your cursor position.

Next, double-click inside the new concept to edit the text and name the concept. This concept represents one of the main ideas you are attempting to document. All main ideas and attributes must have a note detailing a concise description of the concept. For example, a main idea concept named Fixed-Wing Aircraft (FWA) Performance Data Set would be described with the following text: "This is the top-level entity of the Fixed Wing Aircraft Performance model that provides the general performance details."



This is main idea 1's note. It is used for a short concise definition of the concept.

text is shown when the mouse hovers over the node

This is main idea 1's comment. It can be used to wax and wan as necessary to completely describe anything anyone could possibly want to know about a concept.

text is not shown, but can be searched for

Figure 3.1: Notes and Comments on a Concept

To add a note, place the mouse cursor over the concept and right-click. In the ensuing menu, select Add Info.

The dialog box that appears has two separate text entry portions. The box labeled Mouse Over Info is used for the note. Enter your note here and select the OK button.

There is also a text entry box labeled Hidden Info. This space can hold any additional comments about the concept that are above and beyond the scope of the note. Figure 3.1 shows an example where both of these spaces have been filled in.

3.1.3 Create a Concept Attribute. The next step is to create all the necessary descriptive concepts or attributes. Each main idea in the concept map will be described further by child concepts. The following details the creation of these attributes. Repeat the following steps as necessary to create all needed attributes.

Just as with the main idea concepts, all attributes need to have a note detailing a description of the attribute. As an example, one attribute for the FWA Performance Data Set is Total Load Fuel Weight. Its describing note could read: "The weight of the total fuel load for the particular configuration."

Perform the following steps to create a link and an attribute:

1. Position the cursor on top of the recently created concept with the mouse pointer on the two diagonally downward facing arrows as seen in Figure 3.2 and hold down the left mouse button.
2. While continuing to press the left mouse button, drag the cursor to an empty spot on the canvas. A directed arc will appear as shown in Figure 3.3.
3. When you are satisfied with the length of your arc, release the left mouse button. A new concept and link will appear as shown in Figure 3.4.
4. Double-click in the box located on the arc. This names the relationship between the two concepts. All direct links from a main idea concept must be named with the keyword shows, as in Figure 3.5. Finally, the attribute can be then named by double-clicking on the concept.

3.1.4 Create Attribute Descriptors. Details about the attributes must now be captured. To describe the attributes, new concepts need to be created. These concepts are known as attribute descriptors. Select the attribute you wish to add a descriptor for with the mouse pointer. Position your cursor on top of the downward pointing diagonal arrows and drag the cursor to the position where you would like your descriptor to be located. This will create a link from the attribute to a new concept which will be used as the descriptor. Links and concepts can then be named by double-clicking on them.

In order to determine what descriptors to use for each attribute, several questions must first be answered about the attribute:

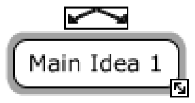


Figure 3.2: Creating link to new Concept

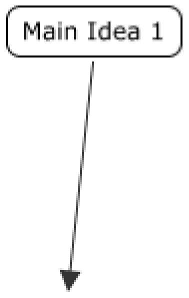


Figure 3.3: Creating link to new Concept, Directed Arc

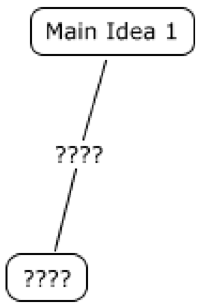


Figure 3.4: Creating link to new Concept, new Concept and link

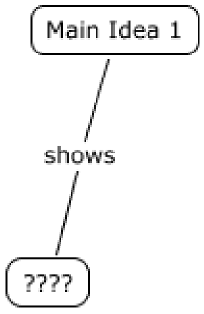


Figure 3.5: Creating link to new Concept, Using keyword *shows*

- *Is it a required portion of the main idea?* This question must be answered by the concept map so the database modeler can force the database to require this attribute as well.

- *Is it unique for every instance of the main idea?* If so, this needs to be identified on the concept map so the database modeler can make use of this attribute as a key in the database table.

- *What data type is it? For instance, is it a STRING of text or an INTEGER?* As a matter of convention, this question should be answered in all capital letters. This is for database informational purposes, as specific data types can require special consideration in the database.

- *How many digits or letters are required to display it correctly?* As with the data type, this information is needed by the database modeler to ensure the data is stored correctly.

- *Is it measurable, and if so, what are the units of measure?* If the attribute is measurable, KPS requires the units of measure; therefore this must be shown on the concept map.

Each of these questions will be answered by the attributes descriptor concepts. In Figure 3.6, you can see that *attribute1* has been labeled as one that *must exist*. Likewise, *attribute2* is of the *INTEGER* data type and will hold up to *10* digits. Any, or all, of the descriptors can be used on any attribute. *Attribute3 is unique*, is *measured in kilograms*, and *is a REAL NUMBER*. Note that the data type for *attribute3* does not include a value for the number of digits. If a value is not specified, a default of 16 digits is assumed.

3.1.5 Linking Keywords. To enable tool supported conversion of the concept map into KPS, we are limiting the scope of the CmapTools application for use by NASIC. Therefore it is necessary to only use specific keywords when naming linking phrases. The following details these keywords.

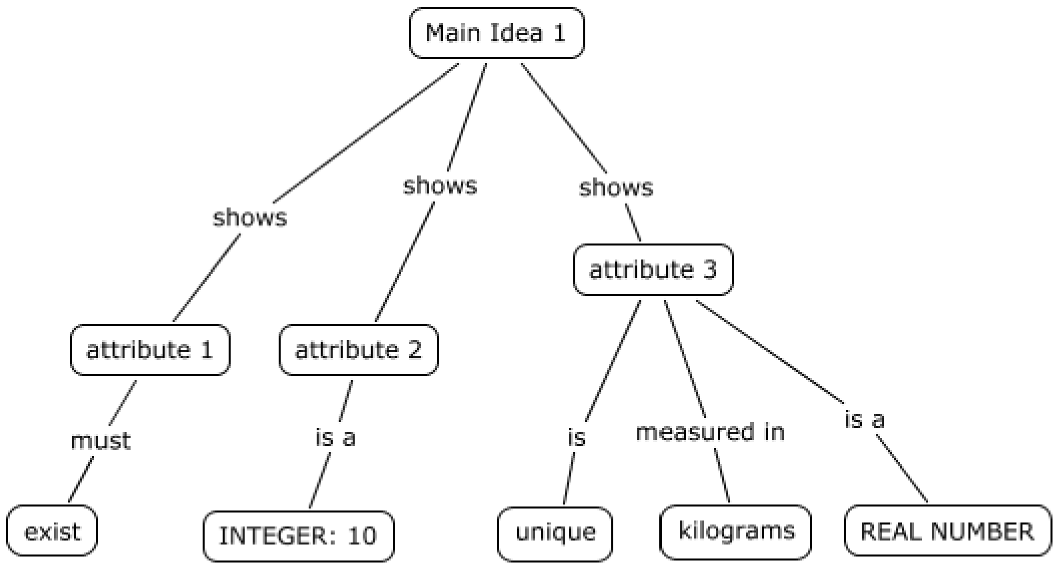


Figure 3.6: Creating attribute descriptors

- shows: The keyword shows was selected as the linking phrase to be used between an entity and its various attributes. Intelligence analysts think in terms of intelligence products and reports so following that line of thinking, each main idea in a concept map would correspond to a section in an intelligence report. Using the keyword shows allows intelligence analysts to visualize the main idea as a section heading that shows various describing concepts about itself. As the semantics of the concept map cannot be checked until the database modeler runs the CMap Conversion for KPS tool any other keyword that attempts to connect a main idea to an attribute will cause that keyword and attribute to be ignored during the conversion process.

The next set of keywords are used to further define the attributes of the concept map.

- has basename: The keyword phrase has basename is used when a main idea can be referenced by a smaller amount of text. For example, the entity Christopher Coryell could have a basename of Coryell.

- has rolename: The keyword phrase has rolename is used when it is desirable to categorize the main idea so that ideas in this subcategory can easily be found in the database at a later date. As an example, the main idea Christopher Coryell could have a rolename of AFIT Student.
- is a: The keyword phrase is a is used to denote the data type of the attribute. It connects the attribute to a concept detailing both the data type and the number of digits of that data type it takes to reference the attribute. Continuing our example, the basename Coryell would be connected to a concept named STRING:7 with this keyword phrase. By convention, the data type is in all capital letters and is followed by a semicolon and an integer denoting the size of the data type, in this case seven letters.
- is: The keyword is is used to specify an attribute that is unique in every instance of a main idea. It must be connected to a concept named unique. This is used to determine whether or not this attribute can be used in the database as either a primary or foreign key.
- must: The keyword must is used to specify an attribute that must be present in all instances of the main idea. It is required to be connected to a concept named exist.
- measured in: The keyword phrase measured in is used when the attribute is measurable. It is connected to a concept that denotes the unit of measure. In our example, if the main idea Christopher Coryell had an attribute named height, the height attribute would be connected via this keyword phrase to a concept named inches.

The final two keywords are used to define relationships between main ideas.

- has: The has keyword is used to reflect a one-to-one relationship between main ideas. This means that only one instance of each main idea can be connected to the other. The main idea Christopher Coryell could be connected to a main idea of Spouse using this keyword.

Table 3.1: Linking Keywords

Keyword	Description
shows	Connects attributes to a main concept
has basename	Connects an attribute to a concept detailing a nickname for that attribute
has rolename	Connects an attribute to a concept used to categorize the attribute
is a	Connects an attribute to a describing data type
is	Connects an attribute to a concept named unique when that attribute is unique to that main concept
must	Connects an attribute to a concept named exist when that attribute must exist
measured in	Connects an attribute to a concept named as the unit of measured; for example, a height attribute would be connected to an inches concept by this linking phrase.
has	Connects two enclosed main concepts when one idea is related to only one of the other
has many	Connects two enclosed main concepts when one concept can be related to many other instances of the second

- has many: The has many keyword phrase is used in a similar manner. It reflects a one-to-many relationship. One idea can contain many instances of the other. Once again using the main idea Christopher Coryell, this keyword could be used to connect to a main idea named Children.

An overview of the keywords can be found in Table 3.1. At this time, these are the only keywords allowed. Any other keywords between main ideas are not accepted and an error will be thrown by the CMap Conversion for KPS tool during conversion if this occurs. If additional keywords become necessary, modifications to the CMap Conversion for KPS tool will have to be accomplished to correctly convert the new keywords and their associated concepts for KPS.

3.1.6 Enclose Main Idea. The finishing step to complete the creation of the main idea is to enclose the main idea with its attributes. This step allows the concept map to hold a great deal of information without overloading anyone who wishes to view your concept map.

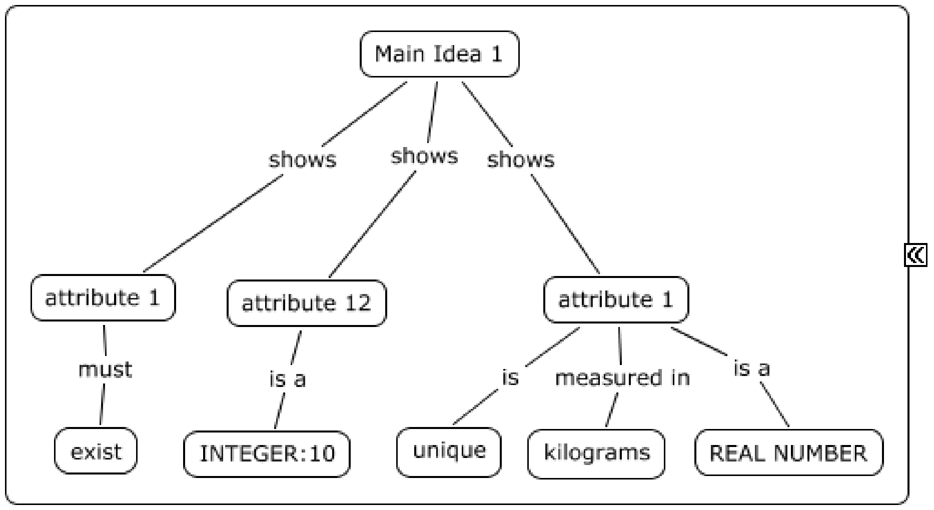


Figure 3.7: Enclosing Main Idea, Enclosure example

The following details the enclosing process:

1. Place your mouse cursor at any of the top or bottom corners of your concept map.
2. Holding down the left mouse button and dragging the cursor towards the main idea content creates a dashed-line box. Continue to drag the cursor until your entire main idea is inside the box.
3. Place your mouse cursor on any part of the selected main idea. Right-click the mouse to bring up the menu. On the menu, select Nested Node, then Create. In Figure 3.7 an example of a nested node is shown.
4. The final step to enclose the main idea is to name the enclosure. The enclosure must have the exact same name as the main idea. To name the enclosure, select the two leftward pointing arrows on the right side of the enclosure as seen in Figure 3.7. This minimizes the enclosure. It can then be named by selecting it with your left mouse button and typing the name. A correctly named enclosure for Figure 3.7 will also be named Main Idea 1.

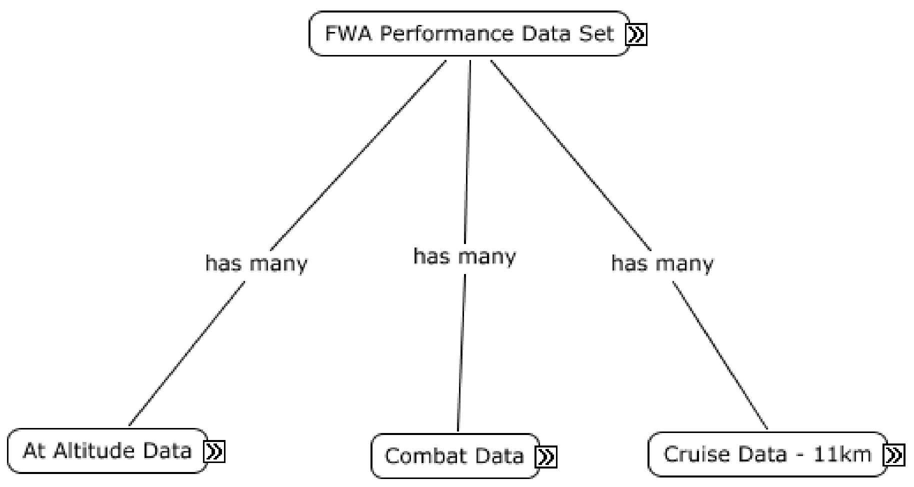


Figure 3.8: FWA concept map example

3.1.7 Connect Main Ideas. The final step in the creation of a concept map is to tie the main ideas together. In most CMaps, there will be a theme tying the main ideas together. As seen in Figure 3.8, the FWA Performance Data Set, which is an enclosed main idea, is linked to three other main ideas. Each instance of the FWA Performance Data Set can be linked to by many instances of Combat Data, At Altitude Data and Cruise Data so they are linked by the keywords has many. These links are created much the same as the links between unenclosed concepts are. First, create all necessary enclosed main ideas. Then select the main idea that will be used as the main theme of the concept map. Position the mouse cursor over the diagonally downward arrows. Hold the left mouse button and drag the directed arc to one of the other main idea concepts. In the new link, use the keywords has if each main theme is linked to one, and only one, of the main idea or the keywords has many if the main theme can be linked to more than one main idea.

When you are satisfied that you have captured all your data on the concept map, save the concept map (select File — Save CMap) and proceed to the next section of this document, Export to XML.

3.2 Export to XML

The second step of the approach is to export the concept map into a format that will be readable by the CMap Conversion for KPS Tool, namely CXL a variant of XML defined in Chapter 2. Fortunately CmapTools has a utility to accomplish this using the following steps:

1. In the window displaying your completed concept map, select File — Export CMap As — XML File...
2. In the file selection dialog that appears, ensure that CXL 1.0 Format appears in the box titled Files of Type. Navigate to the shared directory specified by NASIC/SC and click the Save button.

   Repeat this process until all necessary concept maps have been exported.

3.3 Concept Map Conversion

This final step of the approach will take the XML files generated by the previous step and create the text files needed by the NASIC database modeler for inclusion of the Intelligence product in KPS. This section details the use of the CMap Conversion for KPS tool.

1. First, click the Browse button to the right of the Import File text box.
2. In the dialog that appears, navigate to the concept map XML file repository specified by NASIC/SC and select the appropriate file, then click open.

   The absolute file path will be shown in the Import File text box.
3. In the same manner, select the appropriate directory for the four output files. Select the Browse button to the right of the Output Directory text box. Select the correct directory and click the Open button.

   $\boxed{!}$ Note, when selecting the directory it is imperative that you have only selected the directory and not entered the directory. If you enter the

```
Entity Name#UDP Sequence Within Model#Attribute Name#Attribute ...
Cruise Data - 11km#8#Cruise Mach#Cruise Mach# Cruise Mach#RE ...
Plots#4#Maximum Rate of Climb#Maximum Rate of Climb#Maximum R ...
Combat Data#6#Performance Designator#Performance Designator#P ...
Report Tables#1#Configuration Designator#Configuration Desgina ...
```

Figure 3.9: Elided contents of an example attribute report.txt file.

directory, you'll receive a error message noting that the path you have selected is not a Directory. If you receive this error, select OK to return to the main screen, then ensure that you select the directory via a single click and then click Open as opposed to entering the directory via the double click.

4. After both the Import File and the Output Directory have been correctly set, either click the Import button or select File — Import from the File menu. The tool will begin to import the concept map.

   $\boxed{!}$ All Entities and Attributes are required to have definitions. Due to the fact that CmapTools is a third party vendor application we were unable to force the intelligence analysts to comply with this requirement. Therefore this is checked at this time. Any missing definitions will trigger a dialog informing you of this. If this occurs enter in your choice of text and note the Entity or Attribute name so a definition can be retrieved from the analyst at a later time. This definition will then need to be manually updated in the output file.

   Additionally, relationships between Entities must be either One-to-Zero-One-or-More or One-to-Zero-or-One. If the analyst did not correctly identify the relationship on the concept map (using the has or has many keywords) a different dialog will appear. Enter in the correct relationship and select OK.

   A message denoting successful completion will appear when the tool has finished parsing the document.

   At this point, the concept map data has been converted into the four text files (attribute report.txt, entity report.txt, domain report.txt and relationship report.txt) needed by the database modeler for use by KPS. These files have been generated in the directory selected earlier in this process.

An example of this output is shown in Figure 3.9. The length of the text in the example has been cut off for presentation on the printed page; each line displayed here only shows about one quarter of the total text on that line in the actual file.

IV. Tool Engineering

This chapter describes the design and engineering decisions for the CMap Conversion for KPS tool.

4.1 Cmap Conversion for KPS

The diagram shown in Figure 4.1 details how the tool moves the data. The data is read in from the XML file created from CmapTools. The code in the ParseFile class then parses the XML code and creates the necessary CmapElements. These elements are passed to the Report class which creates the necessary NASIC ERD elements as required for each of the four text files. Finally, the FileOut class creates the text files from the information given to it by the Report subclasses.

The calls between the classes can be seen in Figure 4.2. Following this sequence, the database modeler opens the CMap Conversion for KPS tool via the UI_Start class. After the database modeler enters necessary information, a JButton on the UI is clicked and a new ParseFile is created. After the ParseFile class has finished separating the XML file into CmapElements, it creates a new FileOut object. This FileOut object then creates four reports in turn to complete the necessary text files.

4.1.1 Package Design. During the design of the tool, it was determined that there were three separate operations that this tool would need to accomplish.

1. Provide a graphical user interface for the user.

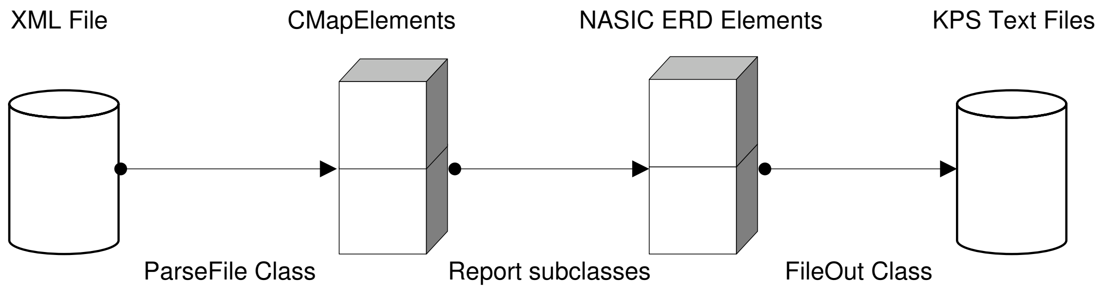


Figure 4.1: Data pathway through CMap Conversion for KPS

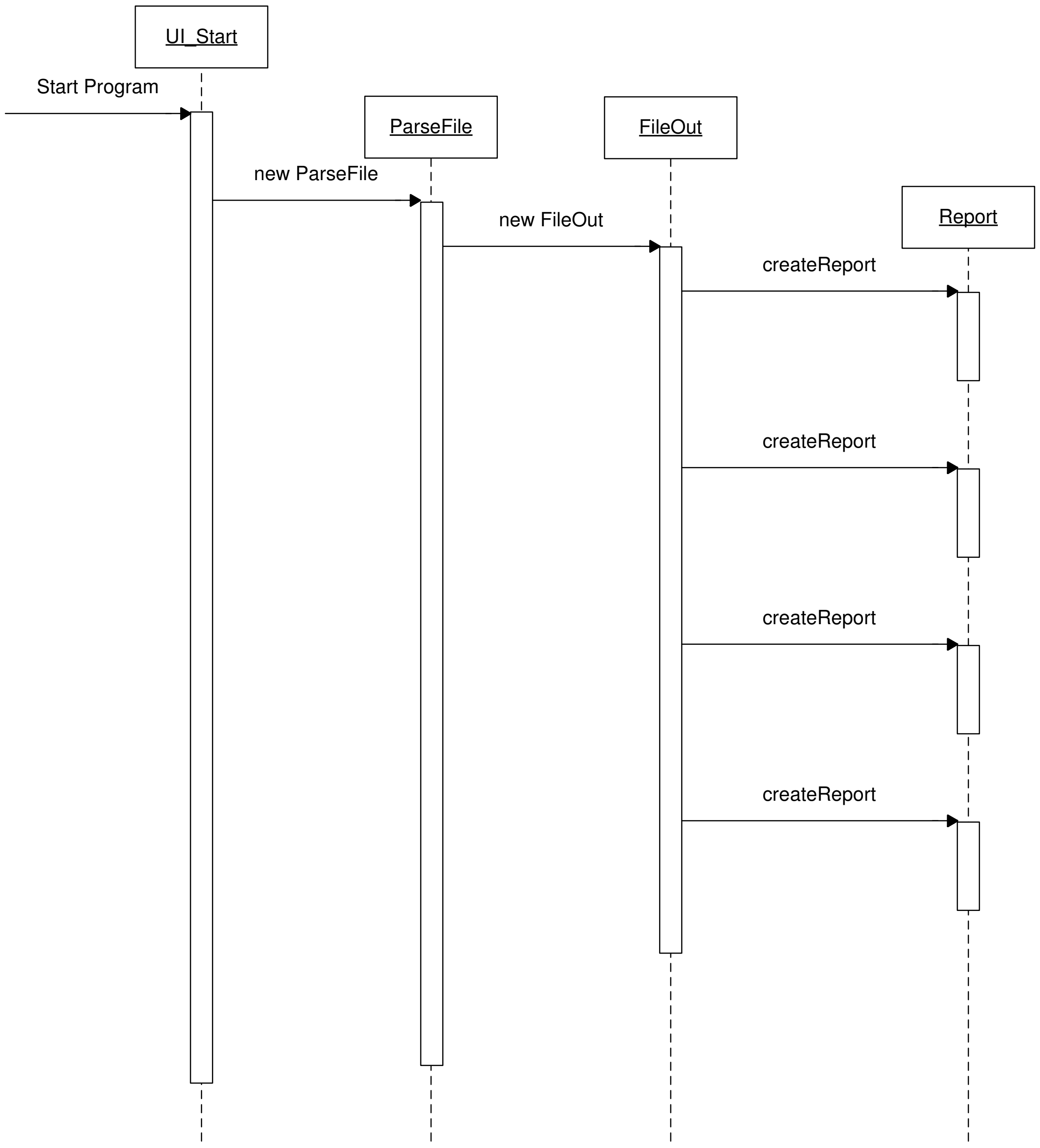


Figure 4.2: Sequence diagram of CMap Conversion for KPS tool

2. Parse the XML file provided by CmapTools into Java Objects representing that same information model.
3. Manipulate files to read input and write output.

With that being the case, we decided to split the source code into three packages; src.ui, src.parse and src.fileOperations.

The src.ui package handles the user interface and is comprised of the UI_Start class.

The src.parse package handles the parsing of the XML file. The main package contains the ParseFile class. Additionally, it contains a sub-package, src.parse.cmapElements. This sub-package contains all the classes necessary to create Java objects representing the various elements of a concept map.

The src.fileOperations package handles the internal creation of the output files as well as the actual writing of the files to disk. The main package contains the FileOut class. This class runs the fileOperations show. It contains the source code to create the output files. This package contains two sub-packages; src.fileOperations.outputFiles and src.fileOperations.fileFilters. The outputFiles sub-package contains the Report class and the four children representing the four different output files. The fileFilters sub-package contains two classes used by the UI_Start class to filter files during input and output file selection.

An overarching class diagram for our code is shown in Figure 4.3.

4.1.2 CMap Elements. As mentioned above, the ParseFile class breaks down the XML document and creates various CmapElement objects. This subsection will describe these elements in more detail. The CmapElement Class and its subclasses can be seen in the Class diagram in Figure 4.4.

CmapElements are the various pieces that the CmapTools application breaks down into. They are propositions, resources, connections, concepts and linking phrases. The Proposition and Resource classes were created for future work as this iteration

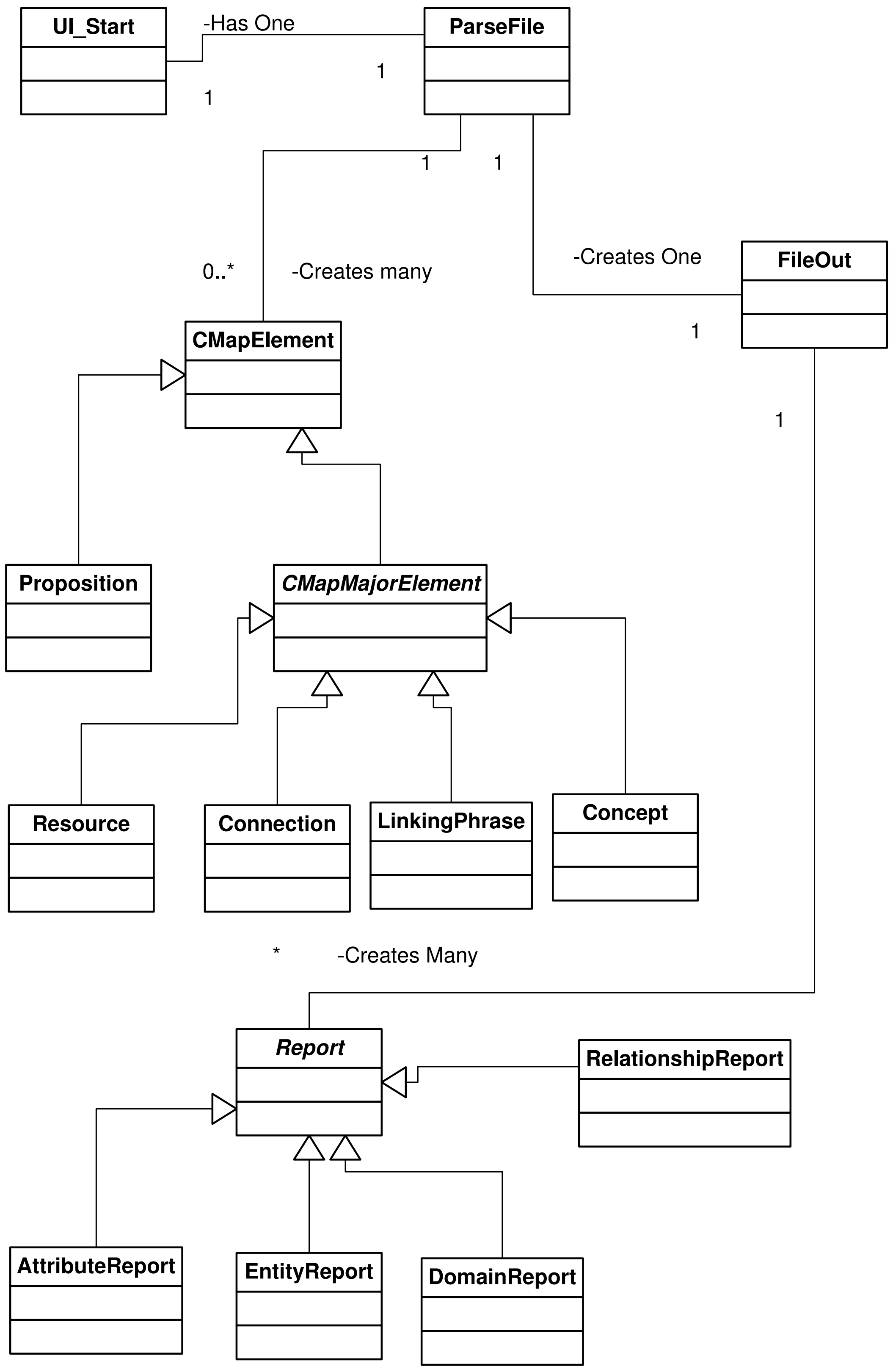


Figure 4.3: Class diagram of entire application, Class names only

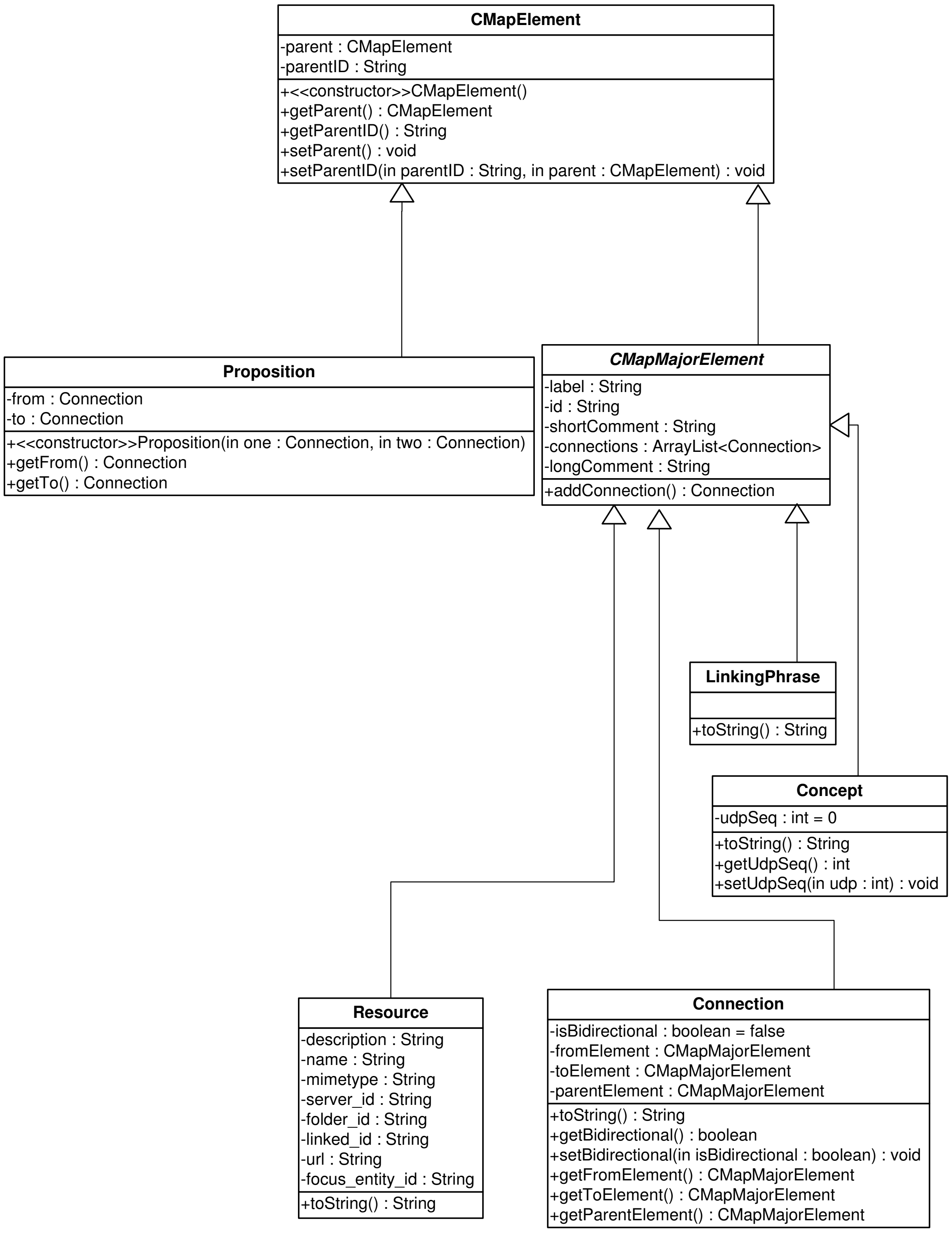

CMapElement
-parent : CMapElement
-parentID : String
+<<constructor>>CMapElement()
+getParent() : CMapElement
+getParentID() : String
+setParent() : void
+setParentID(in parentID : String, in parent : CMapElement) : void
Proposition
-from : Connection
-to : Connection
+<<constructor>>Proposition(in one : Connection, in two : Connection)
+getFrom() : Connection
+getTo() : Connection
CMapMajorElement
-label : String
-id : String
-shortComment : String
-connections : ArrayList<Connection>
-longComment : String
+addConnection() : Connection
LinkingPhrase
+toString() : String
Concept
-udpSeq : int = 0
+toString() : String
+getUdpSeq() : int
+setUdpSeq(in udp : int) : void
Resource
-description : String
-name : String
-mimetype : String
-server_id : String
-folder_id : String
-linked_id : String
-url : String
-focus_entity_id : String
+toString() : String
Connection
-isBidirectional : boolean = false
-fromElement : CMapMajorElement
-toElement : CMapMajorElement
-parentElement : CMapMajorElement
+toString() : String
+getBidirectional() : boolean
+setBidirectional(in isBidirectional : boolean) : void
+getFromElement() : CMapMajorElement
+getToElement() : CMapMajorElement
+getParentElement() : CMapMajorElement

Figure 4.4: Class diagram of CmapElements

of CMap Conversion for KPS does not use this utility of CmapTools. The portion of code that is interesting for us are the other three elements.

- *Concept* – The first class that we will discuss is the Concept class. Any concept created in a concept map through CmapTools will become an object of this class; meaning that ninety percent of what is shown on a concept map becomes one of these objects. The decision on whether the concept refers to a main idea or an attribute is made during the report creation process.
- *LinkingPhrase* – This class is made up of objects created to represent each of the links that are between concepts on the map. This object is very similar to a *Concept* object in code, requiring only a different toString implementation.
- *Connection* – This class represents the connection between CmapMajorElements. Each connection between a Concept and LinkingPhrase is its own Java object. During the report creation process, these connections have to be transversed to determine which NASIC entities each *Concept* object will become.

4.1.2.1 XML Parsing Strategy. Since XML parsing was necessary for this effort, a strategy needed to be selected. The strategy selected is known as SAX. The Simple API for XML(SAX) works incrementally and only parses the portions of the XML document as needed [10]. SAX requires much less memory than other parsing strategies, however any modifications to the structure of the XML document cannot be accomplished in memory as the entire document is not loaded initially. Required changes to the XML document would have to be accomplished manually and then the document would have to be re-parsed. For our purposes, no changes are made to the XML document so this disadvantage can be ignored.

To implement a SAX parser, we looked at three different APIs; Xerces, JAXP, and JDOM.

Xerces, an open source XML parser from Apache, was the first parser we looked at, but integrating it into our project was not easily understood so this option was discarded fairly quickly.

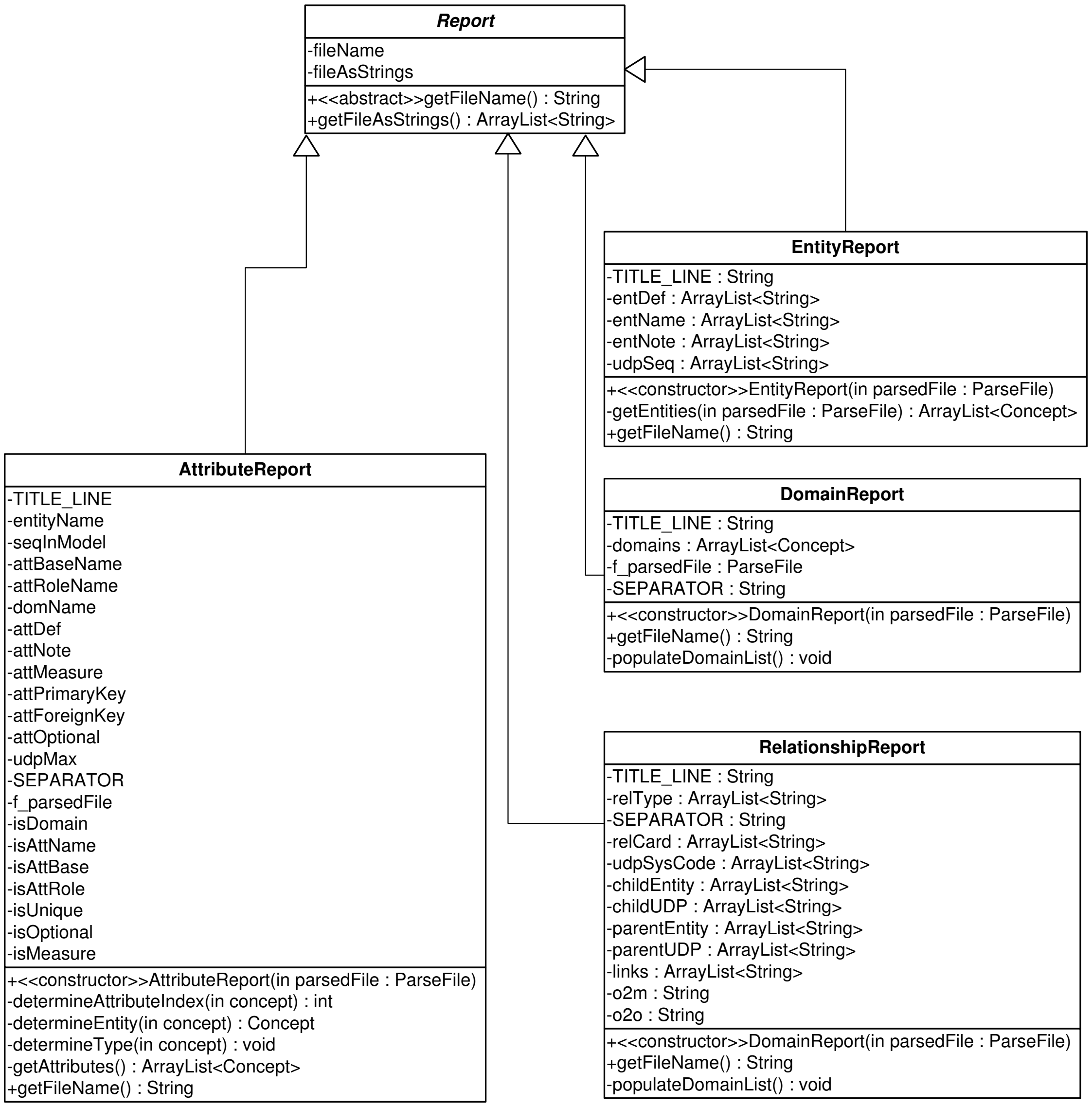
Next JAXP, Sun Microsystem's Java XML parser, was researched but the API was very difficult to understand and we were unable to achieve success using these methods.

Finally, we found an example of JDOM, another open source project created by Jason Hunter, a technologist at Mark Logic Corporation. This library allows XML in a document to be treated much like a Java collection. We were able to open the XML objects created in a debugger and determine how the parsing worked, and customizing it for our purposes followed.

4.1.3 Reports. Each of the four output files are created by an associated Report subclass. As can be seen in the Class Diagram (Figure 4.5), each class parses the CmapElements passed in via the constructor parameter *parseFile*. Each subclass looks for particular characteristics in each CmapElement to determine the elements needed for its output. Each then creates a StringBuilder object for each line of information for the output files. This StringBuilder object is added to the superclasses FileAsStrings (an ArrayList of Strings) to be passed back to the FileOut class for the text files to be written out to disk.

4.1.4 Database Creation. The four output text files our tool creates map to the four tables of our new database. Therefore, each time a concept map is converted, the new database will have a table of entities, a table of attributes, a table of the domains, and a table of relationships.

When a concept map is parsed, it is first broken up into the various CmapElements (concepts, linking phrases, and connections) mentioned previously in this document. Next, an iteration is run to create each of the four tables. Each of the concepts are first visited to determine whether or not they refer to an entity in the database.



Report
-fileName
-fileAsStrings
+<<abstract>>getFileName() : String
+getFileAsStrings() : ArrayList<String>
EntityReport
-TITLE_LINE : String
-entDef : ArrayList<String>
-entName : ArrayList<String>
-entNote : ArrayList<String>
-udpSeq : ArrayList<String>
+<<constructor>>EntityReport(in parsedFile : ParseFile)
-getEntities(in parsedFile : ParseFile) : ArrayList<Concept>
+getFileName() : String
AttributeReport
-TITLE_LINE
-entityName
-seqInModel
-attBaseName
-attRoleName
-domName
-attDef
-attNote
-attMeasure
-attPrimaryKey
-attForeignKey
-attOptional
-udpMax
-SEPARATOR
-f_parsedFile
-isDomain
-isAttName
-isAttBase
-isAttRole
-isUnique
-isOptional
-isMeasure
+<<constructor>>AttributeReport(in parsedFile : ParseFile)
-determineAttributeIndex(in concept) : int
-determineEntity(in concept) : Concept
-determineType(in concept) : void
-getAttributes() : ArrayList<Concept>
+getFileName() : String
DomainReport
-TITLE_LINE : String
-domains : ArrayList<Concept>
-f_parsedFile : ParseFile
-SEPARATOR : String
+<<constructor>>DomainReport(in parsedFile : ParseFile)
+getFileName() : String
-populateDomainList() : void
RelationshipReport
-TITLE_LINE : String
-relType : ArrayList<String>
-SEPARATOR : String
-relCard : ArrayList<String>
-udpSysCode : ArrayList<String>
-childEntity : ArrayList<String>
-childUDP : ArrayList<String>
-parentEntity : ArrayList<String>
-parentUDP : ArrayList<String>
-links : ArrayList<String>
-o2m : String
-o2o : String
+<<constructor>>DomainReport(in parsedFile : ParseFile)
+getFileName() : String
-populateDomainList() : void

Figure 4.5: Class diagram of Report and its subclasses

An entity is created when a concept is found that has no incoming edges. Attributes are then determined by the linking phrase coming out of the entities. Each concept connected to an entity by the linking phrase *shows* creates a new row in the attribute table. This row will then be populated by the concepts connected to the attribute.

To create the domain table, all instances of the linking phrase *is a* are found. The concept this linking phrase is connected to is part of the domain of this database. Care is taken to ensure that there are no duplicate entries in the domain table.

The final table that is created is the relationships. Only top level nested concepts are checked here and they are entered into the table based on the directionality of the linking phrases between the entities as well as the label of the linking phrase. Each linking phrase between entities creates a new row in the database.

The following example will be used to illustrate this process. Consider the concept map shown in Figure 4.6.

The only concepts that do not have an incoming linking phrase are FWA Performance Data Set and Combat Data. These two will become the only entities in the new entity table. Three columns will be populated in the newly created table by the conversion process. The first will be the entity names themselves. The second column is pulled from the definitions of the entities. This is the reasoning behind the requirement of adding them during concept map creation. The final column consists of an identification number created by the sequence in which the concept is visited by the program. So the FWA Performance Data Set row in the table will consist of the entity name, FWA Performance Data Set, the entity defintion, This is the top-level entity of the Fixed Wing Aircraft Performance model that provides the general performance details. and the identification number, in this case the number 2. Table 4.1 shows the created table.

The next pass by the tool creates the attribute table. As mentioned above, each concept whose incoming linking phrase is named shows is selected as an attribute.

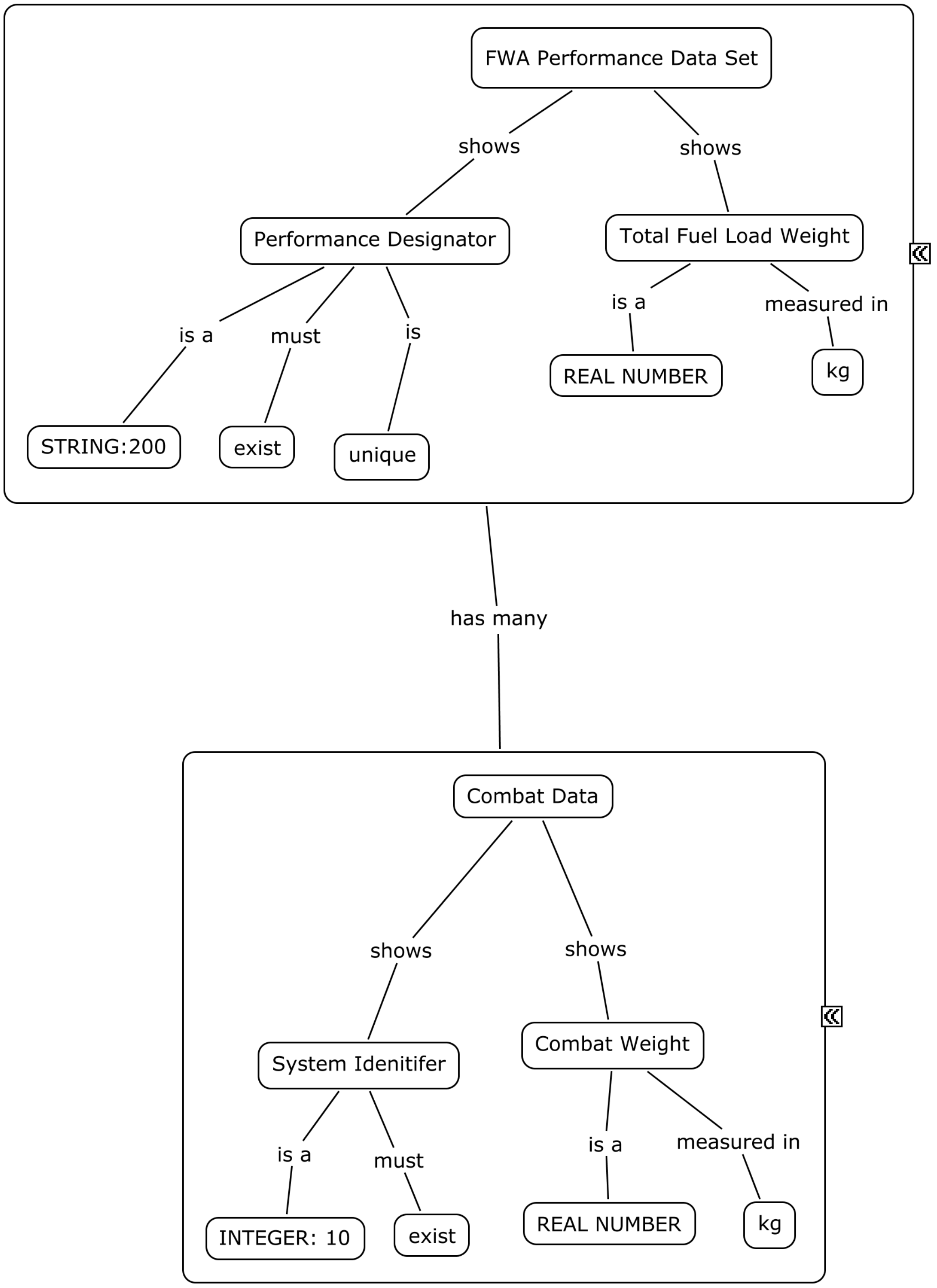


Figure 4.6: Mapping of concept map to database

Table 4.1: Entity table example based on concept map in Figure 4.6

Entity Name	Entity Definition	Sequence Within Model
FWA Performance Data Set	This is the top-level entity of the Fixed Wing Aircraft Performance model that provides the general performance details.	2
Combat Data	This entity provides performance data under combat conditions. Combat Weight is a required attribute and all other attribute values are dependent on this value.	3

A row will be created for each of the following attributes: System Identifier, Total Fuel Load Weight, Combat Weight and Performance Designator. The entity that the attribute is describing is found by following the shows linking phrase back to its source; in the case of Combat Weight the entity would be Combat Data. As in the entity table, a identification number is created based on the sequence in which the conversion tool visits the concept. Other columns created in the attribute table are: Attribute Basename, Attribute Rolename, Domain Name, Attribute Is PK, Attribute Is FK, Attribute Definition, Maximum Length, Unit of Measure and Optional. Each of these is specified using the stylized linking phrases described in Chapter 3. The attribute table created from Figure 4.6 is seen in Table 4.2. Some fields have been left out of the table due to space limitations.

The third table created is the domain table. This table consists of the various data types in use by the concept map. It is entirely populated by concepts that have an incoming linking phrase of is a. Our example has the three domain types; REAL NUMBER, STRING and INTEGER.

An example of the final table created, based on the relationships between entities, is shown in Table 4.3.

Table 4.2: Attribute table example based on concept map in Figure 4.6

Entity Name	Sequence Within Model	Attribute Name	Domain Name	Attribute Is PK	Maximum Length	Unit of Measure	Optional
Combat Data	3	System Identifier	INTEGER	No	10		n
FWA Performance Data Set	2	Total Fuel Load Weight	REAL NUMBER	No	16	kg	y
Combat Data	3	Combat Weight	REAL NUMBER	No	16	kg	y
FWA Performance Data Set	2	Performance Designator	STRING	Yes	200		n

Table 4.3: Entity table example based on concept map in Figure 4.6

Relationship Cardinality	Child Entity Name	Sequence Within Model	Parent Entity Name	Sequence Within Model
One-to-Zero-One-or-More	Combat Data	3	FWA Performance Data Set	2

V. Validation

To validate our claim that our proposed approach is more efficient than the approach in use at NASIC today, we performed a case study using a data set provided by NASIC. Our approach reduced the total time required to create the information model and prepare the text files for importation to the database 89% (down from 40 hours to 4 hours). Additionally, by removing the need for a contractor, our approach has the potential to significantly lower the cost for creation of the information model from \$350K to less than \$150K according to NASIC estimations. The following were the steps taken during our validating case study:

- We received a data set from NASIC; this set corresponds to the knowledge of the intelligence analyst.
- We created a concept map using the data set through CmapTools.
- CMap Conversion for KPS was used to translate the XML produced via CmapTools into KPS text files.

5.1 Data

The example case study is based upon a need for intelligence data on the performance of fixed-wing aircraft (FWA). There are many ideas that need to be conveyed to accurately describe the performance of a FWA. Details of thrust-to-weight ratio, turn and climb rates, and maximum velocity need to be covered. How the aircraft performs in combat must be shown. In addition, the range and mission types for the aircraft are important. All in all, we determined 171 different attributes of a FWA that need to be included in an KPS information model to accurately describe the performance of one FWA.

5.2 Concept Map Creation

Following the steps described in Chapter 3 (creating main ideas, linking them to attributes, enclosing the main ideas and linking main ideas to one another), a full

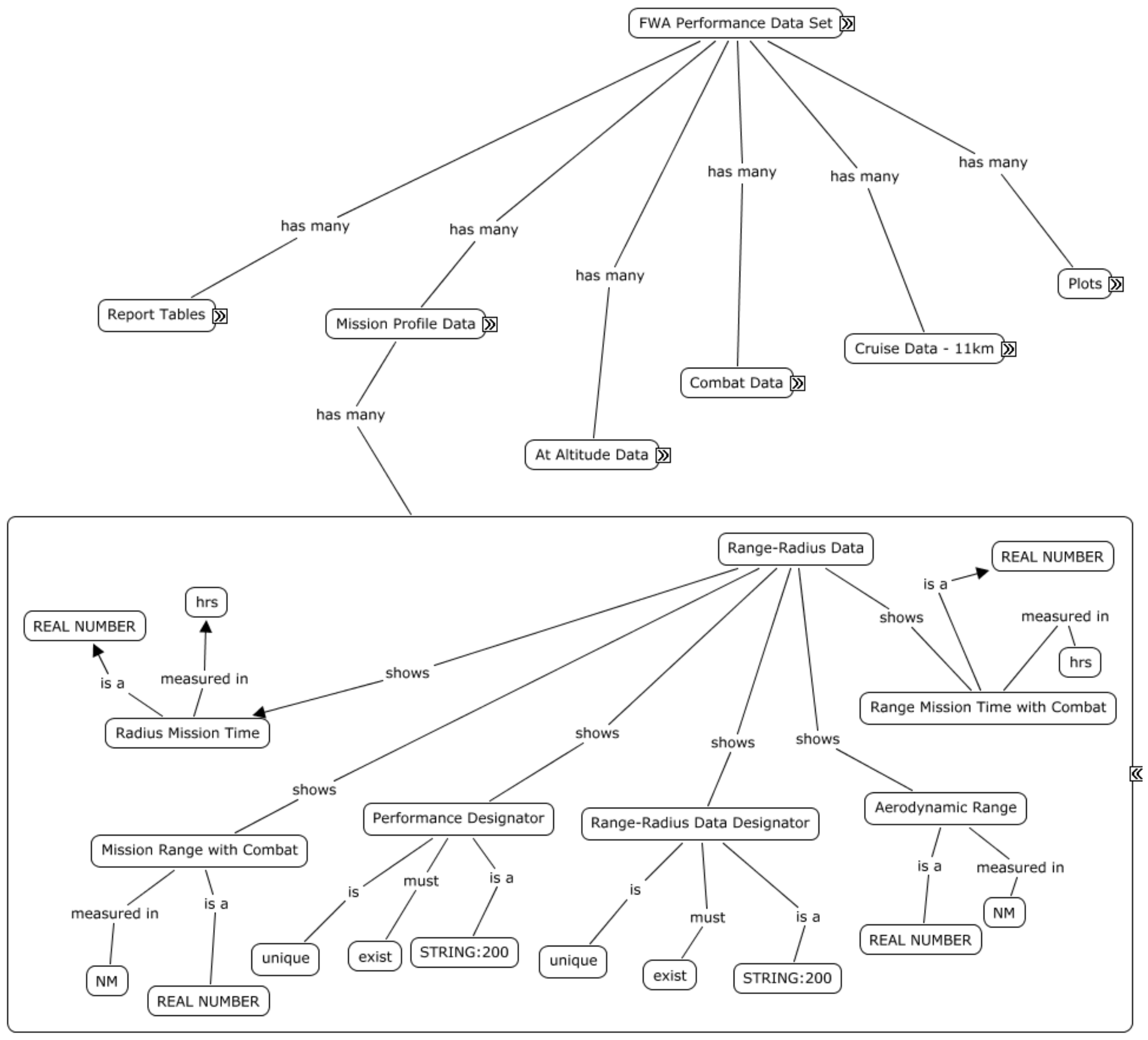


Figure 5.1: Complete FWA Concept Map

concept map for the FWA Performance Data Set was created. The map we created is shown in Figure 5.1. The Range-Radius Data attribute has been expanded to show the full detail of each of the concepts seen in the map. It took 4 hours and 20 minutes to complete the FWA concept map using CmapTools. We believe this duration to be representative of what an intelligence analyst who has been trained to use CmapTools would require to create a similar map.

At this point in the current approach the intelligence analyst would either be in one of the second activities, labeled Analyst explains product information to database modeler in Figure 1.1 (on page 3), or waiting on the availability of the database modeler. If the database modeler was unavailable and a contractor was hired this would add an average of $100K to the cost of production.

5.3 Concept Map Conversion

The final step of the approach is to use the CMap Conversion for KPS tool to convert the concept map created XML into text files. This conversion process was run five separate times. A stopwatch was started after the tool was open, but prior to any operations being performed. The times were 7.8 seconds, 7.1 seconds, 7.8 seconds, 6.8 seconds and 7.3 seconds. These average out to 7.36 seconds to convert the XML files into the necessary text files with the majority of the time spent on navigating the dialog boxes to the location of the XML files. This time spent is negligible when compared with the time it takes to create these text files manually.

At this point, CSIM has resulted in a complete product and is entering the final state in Figure 1.1, Database modeler runs script importing text files to Oracle. 4 hours and 20 minutes into the current approach the intelligence analyst would still be meeting with either the database modeler or the contractor with more than thirty five hours until completion. In all, the CSIM approach was 89% faster as it took 4 hours and 20 minutes as opposed to 40 hours while retaining only 42% of the estimated current cost.

5.4 Threats to Validity

In this section we discuss three threats to the validity of our case study. First, we have assumed that an intelligence analyst is able to create a concept map using CmapTools in approximately the same amount of time as we needed. However, we were unable to verify this with using a NASIC intelligence analyst because, due to scheduling conflicts, the intelligence analyst assigned to work with us was unable to collaborate during this case study.

Second, we have only validated our approach with a single set of test data. It is possible that further real-world intelligence data would be of greater complexity and thus require more time and effort to capture in a concept map or, even worse, not be possible to express in a concept map that our CMap Conversion for KPS tool can automatically convert into a format usable by KPS. However, NASIC has told us that the test data set is of similar complexity to their real-world data.

Third, we were unable to access the actual time-cost of missing information in the intelligence analyst concept maps. With the concept map created as part of our FWA case study, there was no need for the database modeler to correct errors in the map. In real-world CSIM use, there will be maps that require clarification from the intelligence analyst. It is not possible to determine what the time-cost of these clarifications will be due to unknown variables. However, as we note in the next chapter, further integration of CmapTools and CMap Conversion for KPS will help reduce the likelihood of this type error being made by an analyst.

VI. Conclusion

Ensuring that intelligence data is correctly and efficiently recorded and documented in a manner understandable to customers is an ongoing challenge. Due to its expediency and cost savings, the CSIM process for creating KPS information models proposed by this thesis is an improvement over NASIC's current approach.

6.1 Summary of Contributions

This thesis proposes a new tool-supported approach called *CMap Supported Information Modeling* (CSIM) for NASIC intelligence analysts to create KPS information models. CSIM empowers the intelligence analysts to create information models with a reduced need for a database modeler or contractor support. The technical contributions of our work are:

- Development of a stylized concept map that allows an intelligence analyst to systematically specify a new KPS intelligence product. While the concept map tool, CmapTools, was developed by prior work [3], the specification of the *linking keywords* which allow concept maps to be used with KPS is novel.
- Development of the CMap Conversion for KPS tool. A tool to convert our stylized concept map into the text files required by KPS.
- Documentation of our proposed tool-supported approach. We believe this documentation to be critical in the case that CSIM is adopted by NASIC.

6.1.1 Validating Case Study. To validate our claim that our proposed approach is more efficient than the approach in use at NASIC today, we applied the NIMC process to a representative case study, the fixed wing aircraft performance data set, and were able to show a 89% percent decrease in time while also lowering the estimated cost for individual information models by roughly $200K. If the real world gains are similar to these, NASIC could achieve significant cost savings by adopting CSIM.

6.2 Looking Ahead

A limitation of our approach is the inability to check that concept maps follow our *linking keywords* as they are created by the intelligence analyst. We currently find any keyword errors when the database modeler runs our CMap Conversion for KPS tool. This late detection of keyword problems can necessitate further communication with the concept map creator to finish the model. This flaw is unavoidable in the current implementation as CmapTools does not allow customization of the tool.

However, via NASIC's ongoing relationship with IHMC (the creator of CmapTools), it should be possible to integrate CMap Conversion for KPS as an add-on to CmapTools. This would allow for upfront error checking of the information model.

Finally, an additional capability that would be extremely beneficial would be to have CMap Conversion for KPS interface directly with the KPS database through either Java Database Connectivity [8] or Enterprise Java Beans [1]. This would remove the need to generate text files and import them into the database using scripts. In order to support this, a pictorial representation of the database elements should be shown to the database modeler prior to inclusion into the database for their approval.

A direct link between the CmapTools program and the KPS database could also be used to modify concept maps using data from the database. To support this capability, a "reverse" CMap Conversion for KPS tool would have to be written. However, the benefit would be that one tool could be used for for both creating new KPS information models as well as modifying them.

Bibliography

1. Burke, Bill and Richard Monson-Haefel. *Enterprise JavaBeans 3.0.* O'Reilly, 5th edition, 2006.

2. Canas, A.J., R. Carff, G. Hill, M. Carvalho, M. Arguedas, T.C. Eskridge, J. Lott, and R. Carvajal. "Concept maps: Integrating knowledge and information visualization". *Lecture notes in computer science*, 205–219, 2005.

3. Cañas, A.J., G. Hill, R. Carff, N. Suri, J. Lott, T. Eskridge, G. Gómez, M. Arroyo, and R. Carvajal. "CmapTools: A Knowledge Modeling and Sharing Environment". *Concept Maps: Theory, Methodology, Technology, Proceedings of the 1st International Conference on Concept Mapping. Pamplona, Spain: Universidad Pública de Navarra*, 2004.

4. Kostur, P. "Issues in information modeling". *Professional Communication Conference, 2003. IPCC 2003. Proceedings. IEEE International*, 1–5, 2003.

5. Novak, J.D. "The Theory Underlying Concept Maps and How To Construct Them". *Cornell University*, 1982.

6. Novak, J.D. and A.J. Canas. "The Origins of the Concept Mapping Tool and the Continuing Evolution of the Tool". *Information Visualization 5*, 175–184, 2006.

7. Novak, J.D. and D.B. Gowin. *Learning How to Learn.* Cambridge University Press, 1984.

8. Reese, George. *Database Programming with JDBC & Java.* O'Reilly, 2nd edition, 2000.

9. Teorey, T.J. *Database Modeling and Design: The Entity-Relationship Approach.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1990.

10. Wagner, P.J. and T.K. Moore. "Integrating XML into a database systems course". *ACM SIGCSE Bulletin*, 35(1):26–30, 2003.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 22–03–2007	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2005 — Mar 2007

4. TITLE AND SUBTITLE Using Concept Maps to More Efficiently Create Intelligence Information Models	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Christopher E. Coryell, Capt, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Bldg 640 WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCE/ENG/07-03
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Sharon Cain NASIC/SC 4180 Watson Way WPAFB, OH 45433 Sharon.Cain@wpafb.af.mil, (937) 257-2241	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis proposes a tool-supported approach to allow intelligence analysts to create information models for the National Air and Space Intelligence Center with almost no database modeler or contractor support. Our approach allows analysts to express an information model as a concept map, a analyst-understandable model of an intelligence domain. An existing tool, CmapTools, supports the analyst-in-the-loop process of concept map creation. A completed concept map is automatically converted database form by a prototype tool, called Cmap Conversion for KPS, created as part of this work. We restrict, to a reasonable degree, how analysts express concept maps within CmapTools to ensure that automatic conversion is possible.

15. SUBJECT TERMS

Concept Maps, Information Models, NASIC, KPS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lt Col T.J. Halloran
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	45	(937) 255–3636, Timothy.Halloran@afit.edu